# TMO—The Federated Ontology of the TrendMiner Project

**Hans-Ulrich Krieger, Thierry Declerck**

German Research Center for AI (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{krieger|declerck}@dfki.de

**Abstract**

This paper describes work carried out in the European project TrendMiner which partly deals with the extraction and representation of real time information from dynamic data streams. The focus of this paper lies on the construction of an integrated ontology, TMO, the TrendMiner Ontology, that has been assembled from several independent multilingual taxonomies and ontologies which are brought together by an interface specification, expressed in OWL. Within TrendMiner, TMO serves as a common language that helps to interlink data, delivered from both symbolic and statistical components of the TrendMiner system. Very often, the extracted data is supplied as quintuples, RDF triples that are extended by two further temporal arguments, expressing the temporal extent in which an atemporal statement is true. In this paper, we will also sneak a peek on the temporal entailment rules and queries that are built into the semantic repository hosting the data and which can be used to derive useful new information.

**Keywords:** multilingual multifaceted integrated ontology; temporally-changing information & temporal entailment; web harvesting & migration rules.

## 1. Introduction

This paper describes work carried out in the European project TrendMiner (`www.trendminer-project.eu`) which deals, in part, with the extraction and representation of real time information from dynamic data streams, such as blogs, twitter, newswires, and wikis. Besides the dynamic nature and the huge amount of data being processed, TrendMiner addresses two large case studies, viz., *assisting financial investing decisions* (e.g., by harvesting stock exchange Web pages) and an *EU-wide tracking of political views, trends, and politician popularity over time*.

The **focus of this paper** lies on the construction of an integrated ontology, **TMO**, the TrendMiner Ontology, that has been assembled from several independent multilingual taxonomies and ontologies which are brought together by an interface specification, expressed in OWL (McGuinness and van Harmelen, 2004).[1] Within TrendMiner, TMO serves as a common language that helps to interlink data, delivered from both symbolic and statistical components of the TrendMiner system.

Very often, the extracted symbolic data from webpages is supplied as *quintuples*, RDF triples that are "annotated" by two further temporal arguments, expressing the temporal extent in which an atemporal fact holds (essentially, an extension of the plain N-Triples format; see (Grant and Beckett, 2004)). In order to store such quintuples, they are either transformed into a set of semantic-preserving triples when stored in a triple repository like OWLIM (Kiryakov et al., 2005), applying, e.g., W3C's N-ary relation encoding scheme (Hayes and Welty, 2006), or can be utilized immediately, when recorded in an N-tuple repository, such as *HFC* (Krieger, 2013).

In this paper, we will also sneak a peek on the temporal entailment rules (Krieger, 2012) and queries that are built into one of the semantic repository hosting the data and which can be used to derive useful explicit information. This includes identifying companies operating in similar areas, monitoring data for unusual events, or making knowledge about people explicit.

We will also describe how information is harvested from company websites (resulting in *company snapshots* = sets of quintuples) and how this data is made compatible with the ontology schema, using so-called migration rules. Finally, we will look more closely on the multilingual information encoded in some of our sub-ontologies and how this might help during automatic ontology alignment.

## 2. Ontologies

Overall, **TMO** consists of 18 sub-ontologies, **sixteen** of which are truly independent and do not have knowledge of one another. **Two** further ontologies, called *IF* and *XEBR2XBRL*, bring them together through the use of interface axioms, using axiom constructors, such as rdfs:subClassOf and owl:equivalentProperty, or by posing domain and range restrictions on certain underspecified properties. It is worth noting that across the ontologies, each property has been cross-classified as being either *synchronic*, i.e., property instances staying constant over time, or *diachronic*, i.e., changing over time (Krieger, 2010). This property characteristic can be used, amongst other things, to check the consistency of a temporal ABox or as a distinguishing mark in an entailment rule (see, e.g., Sections 3.4. and 3.5.).

Let us introduce the **18 sub-ontologies of TMO** (see Figure 1) and then focus on a few selected **highlights** in Sections 2.1–2.5.

1. *BIO* (biographical facts about people and events)
2. *CFI* (ISO's classification of financial instruments)
3. *DAX* (stock exchange: Deutscher Aktien Index)
4. *DC* (one concept, three properties from Dublin Core)

[1] The ontologies are publicly available for open research and to other institutions upon request; see `www.dfki.de/lt/onto/`.

5. *EN* (stock exchange: NYSE Euronext)
6. *GICS* (Standard&Poor's/MSCI industry sector classification)
7. *ICB* (Dow Jones/FTSE industry sector classification)
8. *IF* (most of the interface axioms)
9. *LOGIC* (modalized propositions; used by *SENT*)
10. *NACE* (EU/UN industry sector classification)
11. *OP* (opinion: extends the MARL ontology)
12. *POL* (political facts about people and events)
13. *SENT* (sentiment, uses *LOGIC*)
14. *SKOS* (SKOS relations applicable to classes)
15. *SOC* (translation of TheSoz/GESIS sociology thesaurus)
16. *TIME* (distinction: synchronic/diachronic properties)
17. *XEBR* (XBRL Europe Business Registers)
18. *XEBR2XBRL* (interfacing XEBR and local XBRL jurisdictions)

Even though ABox data (populated instances) usually come with a temporal extent, the TBoxes and RBoxes of the ontologies are not equipped with temporal information, thus still being represented as triples. For instance, we can not state that an URI represents a class at a certain time and a property at a different time. Or that a class is a subclass of another class for only some amount of time. Thus TBox and RBox of the integrated ontology represent knowledge that is true at any time, so there is no need to equip them with a fourth and fifth temporal argument. This quality gives rise to the use of ontology editors, such as Protégé, for manually constructing the TBoxes and RBoxes of some of our ontologies.

We note here that most ontologies are multilingual in that both classes, properties, and predefined instances are assigned multiple and multilingual labels or even longer definitions in different languages, making use of the annotation properties rdfs:label, skos:prefLabel, and skos:altLabel, together with an additional annotation property which we called rdfs:definition.

TMO, as such, can be seen as a mid-level ontology which takes a liberal stand against top-level ontologies (such as SUMO or DOLCE) or against ontologies which try to describe specific aspects of an agent (e.g., a company) on a very general level (e.g., the Registered Organization Vocabulary). Nevertheless, it is clear that concepts and properties from these ontologies can be interlinked to TMO via interface axioms, as described in Section 2.5..

## 2.1. *BIO*

*BIO* is used to represent biographical facts about people's lives. The ontology comes with a tripartite structure of the following pairwise disjoint classes, subclasses of the most general class Entity:

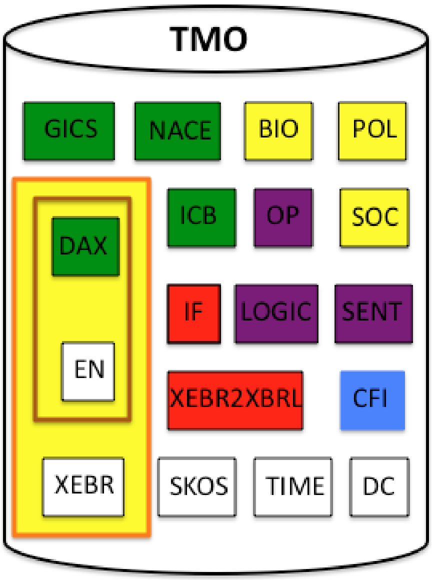

Figure 1: The TMO ontology consists of 18 sub-ontologies overall. The color encoding refers to ontologies focussing on models of *people's private and public lives* (**yellow**), *sentiment/opinion* (**purple**), *industry sector classification* (**green**), *stock exchange* (**brown** rectangle), *financial reporting* (**orange** rectangle), *financial instruments* (**blue**), and *interface* (**red**). As can be seen from the picture, some of the ontologies even model several aspects of a domain; e.g., *DAX* alone deals with industry sector classification, reporting, description of stock exchange listed information, and people who are either key executives or shareholders of a company.

- Abstract. An Abstract thing can manifest in a Happening, whereas a Happening might lead to an Entity. Abstract can be used to describe *literal* concepts, such as *activities, ideas, dreams, the life*, etc. Abstractions can further manifest in real-world Happenings. The outcome of a Happening can be virtually everything.
- Happening. Happenings are either static atomic Situations or dynamic decomposable Events. A Happening keeps its own startDate and endDate, involves Agents, and happensAt a Location. An Event startsWith, continuesWith, and endsWith a Happening, thus this modelling structure can be used to model simple processes.
- Object. The subclass Agent makes a fundamental distinction between Person, Group, and political State. People might be involved in a happening or learn about it (property owns), thus being aware of a happening using property isAwareOf. Since isAwareOf is a diachronic property, we can easily model (using the temporal extent) that awareness might turn into oblivion.

Assertional knowledge (i.e., ABox relation instances) in *BIO* is usually encoded as quintuple with the notable exception of instances of Happening which encapsulate their own starting and ending time (see above).

## 2.2. *EN*

The Euronext ontology *EN* does not come up with its own industry classification, but makes use of *ICB* (see below). *EN*, in part, reduplicates the stock exchange ontology *DAX*, but uses different names for classes and properties. However, more financial numbers are given here, even for three succeeding year.

Let us take an example. *Credit Agricole*'s revenues for 2012, 2011, and 2010 are listed today as 16,315,000, 20,783,000, and 20,129,000 Euros. Even at the end of this year, these numbers will be the same. However in 2014, the number for 2010 will no longer be listed, but instead, we will then find only numbers for 2013, 2012, and 2011.

Clearly, we do *not* want to extend the ontology with new property names every time a new business year starts, thus we must avoid properties such as hasRevenue2013, hasRevenue2012, hasRevenue2011, etc.

In order to address this and to properly represent the numbers against the varying date when the information was harvested from the companies websites, we use a simple "trick" here: we always use exactly the three properties hasRevenue-1, hasRevenue-2, and hasRevenue-3. The hyphen character - now should be interpret as a minus sign −, thus, e.g., the value stored under hasRevenue-2 actually refers to the revenue *two* (2) years ago *relative* to the actual business year when the company snapshot was taken (the business year is stated elsewhere).

### 2.3. Industry Sectors: *ICB*, *DAX*, *GICS*, and *NACE*

ICB, the *Industry Classification Benchmark*, is an industry classification schema used worldwide at many places (e.g., NYSE in New York), developed by Dow Jones and FTSE. Euronext (as operated by NYSE) makes use of ICB's four-level deep classification in its description of financial titles. Given the ICB terminology stated in Microsoft Excel documents, we have auto-generated an OWL ontology *ICB* that arranges the 186 industry sectors in a subsumption hierarchy.

*ICB* is connected to *EN* through an axiom from *IF* and comes up with an informal sector description for English, German, and Spanish, together with a further multilingual "definition" of the most specific concepts. In order to address these definitions on the class level properly, we make use of a further annotation property which we have called rdfs:definition.

TMO has integrated further industry classification taxonomies which partially overlap with *ICB* (see also Section 3.1.), but establish different views on industry sectors, viz.,

- *DAX* (origin: Deutsche Börse): comes with its own two-level deep sector classification; 81 classes; German and English labels.
- *GICS* (origin: Standard & Poor's and MSCI): four-level deep sector classification; 268 classes; 10 languages.
- *NACE* (origin: EU and UN): four-level deep sector classification; 997 classes; English, German, and Italian labels at the moment.

The top-level sector classes of the four industry classification ontologies have been manually identified using OWL's equivalentClass axiom constructor (see also Section 2.5.1.).

### 2.4. *OP*

This opinion ontology is based on the *Marl* ontology, described in (Westerski et al., 2011). Even though some of the property names would have been labelled differently by us (e.g., using hasTarget instead of describesObject in order to be compatible with opinion mining terminology), we have not alter the original property names.

We have, however, made some adjustments to *Marl* and have added further properties as described below:

- extractedFrom is now a datatype property, mapping to xsd:anyURI;
- we have added the object property hasHolder (range: underspecified);
- we have added the datatype property holdersTrust (range: xsd:double);
- we have added the datatype property utteredAt (range: xsd:dateTime);
- we have declared certain properties to be functional;
- we have defined the range type for already-existing properties.

Some of the original properties (e.g., describeFeature) as well the new property hasHolder are not assigned a range class in *Marl*. In order to constrain these properties further, we recommend (as we have done in TMO) to add further interface axioms, e.g., *the holder of an opinion is an agent/person* (see Section 2.5. below).

We have furthermore classified all properties in the opinion ontology as diachronic properties. This has the advantage that such a treatment makes it easy to see how an opinion evolves/changes over time. Note that this evolution mostly happens for aggregated opinions, but might even happen for information related to a single opinion, say, the holders' trust changes within a longer period of time.

### 2.5. *IF*

As already explained, the interface ontology *IF* interlinks the 16 sub-ontologies through manually specified interface axioms. To achieve this, *IF* makes use of *DC*, *SKOS* and *TIME*, but mostly utilizes the standard axioms constructors from RDFS and OWL, together with domain and range restrictions, i.e.,

- owl:equivalentClass
- rdfs:subClassOf
- owl:equivalentProperty
- rdfs:subPropertyOf
- owl:sameAs
- rdfs:domain
- rdfs:range
- rdf:type

Here are some examples, using description logics (DL) syntax.

### 2.5.1. Classes and Properties

dax:Company, en:Company, nace:IndustrySector, and gics:GICS can be used interchangeably; xebr:Report is a subclass of dc:Resource; the properties dax:portrait and en:activity are equivalent (DL syntax):

$\text{dax:Company} \equiv \text{en:Company}$
$\text{dax:Company} \equiv \text{gics:GICS}$
$\text{dax:Company} \equiv \text{nace:IndustrySector}$
$\text{xebr:Report} \sqsubseteq \text{dc:Resource}$
$\text{dax:portrait} \equiv \text{en:activity}$

Note that the transitivity of owl:equivalentClass guarantees that dax:Company, en:Company, gics:GICS, and nace:IndustrySector are belonging to the same equivalence class.

### 2.5.2. Domain & Range Restrictions and Typing

XEBR reports are linked to companies via the diachronic functional object property if:hasReport; the holder of an opinion is an agent:

$\top \sqsubseteq \forall \text{if:hasReport}^{-} . \text{dax:Company}$
$\top \sqsubseteq \forall \text{if:hasReport} . \text{xebr:Report}$
$\text{if:hasReport} : \text{owl:FunctionalProperty}$
$\text{if:hasReport} : \text{owl:ObjectProperty}$
$\text{if:hasReport} : \text{time:DiachronicProperty}$
$\top \sqsubseteq \forall \text{op:hasHolder} . \text{bio:Agent}$

The last axiom together with
$\text{bio:Person} \equiv \text{pol:Person}$
gives us the possibility to talk about, e.g., journalists and their opinions, due to the following subclass axioms, specified in *BIO* and *POL*, resp.:
$\text{bio:Person} \sqsubseteq \text{bio:Agent}$
$\text{pol:Journalist} \sqsubseteq \text{pol:Person}$

### 2.5.3. Meta-Modelling

The class DiachronicProperty is a subclass of rdf:Property; the property partOf is a property connecting OWL classes, *not* instances:

$\text{time:DiachronicProperty} \sqsubseteq \text{rdf:Property}$
$\top \sqsubseteq \forall \text{xebr:partOf}^{-} . \text{owl:Class}$
$\top \sqsubseteq \forall \text{xebr:partOf} . \text{owl:Class}$

## 3. Rules

This section presents some showcases that involve individual ontologies, interlinking axioms, and domain-specific queries and entailment rules.

### 3.1. Finding Competitors Across Stock Exchanges

Characterizing a company against an industry sector classification is an extremely important showcase which involves finding competitors of a company that work in a similar field. We attack this problem in two ways. *Firstly*, we have established manual mappings between sectors from different classification schemes, such as (all four classes talk about *financial institutions*)

$\text{icb:ICB8300} \equiv \text{nace:nace\_64.1}$
$\text{icb:ICB8300} \equiv \text{dax:Banks}$
$\text{icb:ICB8300} \equiv \text{gics:GICS4010}$

*Secondly*, we are currently trying to automatically align the multilingual labels of the *NACE*, *ICB*, *DAX*, and *GICS* classes in order to establish a unified sector classification across the four ontologies (see also Section 5.).

Since the mappings connect industry sectors across *different* stock exchanges, querying for companies of type dax:Banks will automatically yield companies classified as icb:ICB8300, nace:nace_64.1, or gics:GICS4010. Here is an example involving competitors of *Deutsche Bank*, making use of the query language in *HFC* (Krieger, 2013) to access quintuples in the `WHERE` clauses:

```
SELECT DISTINCT ?competitor
WHERE ?db dax:name "Deutsche Bank" ?s ?e
      ?db rdf:type ?type ?s ?e
      ?competitor rdf:type ?type ?s2 ?e2
FILTER ?db != ?competitor
```

### 3.2. Monitoring Unusual Events

"Unusual" events refer to important changes that have happened in a company or in a person's life, say, the replacement of a CEO or the change of the *transparency standard* (a company can not adhere to more than one standard at the same time). If the latter happens, a rule can leave a *memento* in the repository that can be queried later. Here is an example, making use of *HFC*'s rule language:

```
?c dax:transparencyStandard ?ts1 ?s1 ?e1
?c dax:transparencyStandard ?ts2 ?s2 ?e2
->
?mem rdf:type if:Memento ?e1 ?s2
?mem if:changeStandard ?c ?ts1 ?ts2 ?e1 ?s2
@test
?ts1 != ?ts2
DTLess ?s1 ?s2
@action
?mem = MakeUri ?c ?e1 ?s2 ?ts1 ?ts2
```

The predicate (`@test`) `DTLess` guarantees that `?s1` is smaller than `?s2` (both variables will bind XSD atoms of type `dateTime`). The action (`@action`) `MakeUri` deterministically generates a new URI from its input arguments `?c`, `?e1`, `?s2`, `?ts1`, and `?ts2`. This URI then is used on the RHS of the rule to store the relevant information, viz., the company, the different standards, and the period in which the change has happened.

### 3.3. Making Knowledge About People Explicit

The below depicted *HFC* rules unveil simple, but still useful knowledge, e.g., *people from political governance (chancellor, judges, ministers) are also regarded as political figures* (note the fourth and fifth temporal arguments):

```
?p rdf:type pol:PoliticalGovernance ?s ?e
->
?p rdf:PoliticalFigure ?s ?e
```

*People from a sub-organization of a party are also party members.* Note that `Max2` and `Min2` below implement an intersection of the two temporal intervals $[b_1, e_1]$ and $[b_3, e_3]$; see (Krieger, 2012):

```
?x rdf:type pol:Party ?b1 ?e1
?x pol:hasOrganization ?y ?b2 ?e2
```

```
?y pol:hasMember ?p ?b3 ?e3
->
?x pol:hasMember ?p ?b ?e
@action
?b = Max2 ?b1 ?b3
?e = Min2 ?e1 ?e3
```

### 3.4. Domain-Independent Temporal Entailment

In (Krieger, 2012), we have presented a temporal extension of the Hayes (Hayes, 2004) and ter Horst (ter Horst, 2005) entailment rules for RDFS and OWL and have shown that *temporal* reasoning and querying with triples is extremely complex, expensive, and error-prone when compared with quintuple-based representations (see also Section 6.).

For instance, we have complemented the original rule rdfp1 in (ter Horst, 2005) dealing with object properties by a new rule that also addresses datatype properties. Let us start with the assumption that the object is either a URI or a blank node, exactly what the original rule encodes in its *where* condition:

```
?p rdf:type owl:FunctionalProperty
?p rdf:type owl:ObjectProperty
?p rdf:type time:DiachronicProperty
?x ?p ?y ?s1 ?e1
?x ?p ?z ?s2 ?e2
->
?y owl:sameAs ?z
@test
IntersectionNotEmpty ?s1 ?e1 ?s2 ?e2
```

The IntersectionNotEmpty predicate in the test section (@test) guarantees that we only *identify* ?y and ?z on the RHS in case the temporal extent of $p(x,y)$ and $p(x,z)$ has a *non-empty intersection*:

```
IntersectionNotEmpty start1 end1 start2 end2 ≡
  start := max(start1, start2)
  end := min(end1, end2)
  return (start ≤ end)
```

Thus a single overlapping observation leads to a *total* identification of ?y and ?z (at all times!), so the sameAs statement need not be equipped with temporal information. If both observations, however, do talk about *different* non-intersecting times, it makes perfect sense that ?y and ?z need *not* be equal, even though ?p is a *functional* property (good example: marriedWith relation).

Let us now focus on the second rule, dealing with functional *datatype* properties.

```
?p rdf:type owl:FunctionalProperty
?p rdf:type owl:DatatypeProperty
?p rdf:type time:DiachronicProperty
?x ?p ?y ?s1 ?e1
?x ?p ?z ?s2 ?e2
->
?x rdf:type owl:Nothing ?s ?e
@test
?y != ?z
IntersectionNotEmpty ?s1 ?e1 ?s2 ?e2
@action
?s = Max2 ?s1 ?s2
?e = Min2 ?e1 ?e2
```

If two non-identical atoms are defined on a property, the above rule signals a problem by assigning the bottom type owl:Nothing to the URI in the first place of the tuple. Since $p(x,y,s_1,e_1)$ and $p(x,z,s_2,e_2)$ come with a duration, the type assignment to ?x only holds for the intersection of the two intervals $[s_1,e_1]$ and $[s_2,e_2]$, computed by Max2 and Min2.

### 3.5. Merging Temporal Extents

The next rule turns two quintuples which coincide in subject, predicate, and object position and which share a non-empty temporal intersection into a larger unit. Consider, for instance, the *ceoOf* relation between a person $p$ and a company $c$. Commonsense dictates that

$$ceoOf(p,c,s1,e1) \wedge ceoOf(p,c,s2,e2) \wedge s_2 \leq e_1$$

should entail

$$ceoOf(p,c,s1,e2)$$

Since only *diachronic* properties are supposed to change over time, we add a further typing constraint and *quantify* over the property position ?p in the below *HFC* rule:

```
?p rdf:type time:DiachronicProperty
?c ?p ?v ?s1 ?e1
?c ?p ?v ?s2 ?e2
->
?c ?p ?v ?s1 ?e2
@test
DTLess ?s2 ?e1
```

## 4. Instance Data

We already mentioned in the introduction that instance data is often delivered as quintuples (relational fluents), i.e., binary relation instances that have been extended by two further temporal arguments, representing the temporal interval (starting and ending time) in which the relation instance is true. In the past, this was realized by *harvesters*, standalone Java programs that produce ABox data, compliant with the ontologies. Since the HTML representation of the information changed rapidly and sometimes drastically (especially for stock exchange data), we were forced to change the program code of the harvesters on the input side over and over again. And in case our ontologies were modified, the output of the harvesters need to be adapted too.

In order to change this unsatisfying procedure and to abstract from the *imperative* program code, we decided to use a dedicated *syntactic web scraper* instead, in order to create the extended RDF datasets (Kumar Nedunchezhian, 2013). The use of this scraper has two important advantages, viz.,

1. the information in which we are interested in is specified *declaratively*, and
2. the output is agnostic against the target ontologies.

Let us focus on these two aspects and see how we finally address the target ontologies by using the *HFC* reasoner from above as a succeeding migration service.

Describing *what* to extract from an HTML page is specified in the scraper by defining declarative rules that address the different ways information is specified on a web page, e.g.,

- one key, one value;
- one key, many values (table with a horizontal header);
- more than one key and values (table with horizontal and vertical labels).

Given the source HTTP address where the information is located, the scraper then produces N-tuples which come with brand-new URI anchors (generated from the heading name and Unix's epoch time) and that use the key(s) and value(s) from the web page as elements of the N-tuples. Since the information was taken in a moment of time, the starting and ending time of the N-tuple coincide.

For instance, the address for *adidas* can be found on the DAX page `http://www.boerse-frankfurt.de/en/↩ /equities/adidas+ag+DE000A1EWWW0/company+data` under *heading* `Address`, where the actual address is listed there again under *key* `Address`, together with other information, such as the phone number. Given this input, the scraper then generates the following N-tuples (we have slightly shortened the URIs for better readability):

```
scrap:adidas_AG_1382726391024
scrap:HEAD_Address
scrap:Address_1382726391026
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .

scrap:Address_1382726391026
scrap:KEY_Address
"adidas AG
 Adi-Dassler-Strasse 1
 91074 Herzogenaurach
 Deutschland"
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .

scrap:Address_1382726391026>
scrap:KEY_Phone
"+49 (0)9132 84 - 0"
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .

.....
```

Now, in case the input representation changes, we have to update the declarative scraping rules, instead of modifying the Java program code (which we find is much easier).

The question now remains how we guarantee that the N-tuples, produced by the scraper, are compliant with the quintuples for the target ontologies, as

1. the generated property names differ,
2. the values require some "polishing",
3. information from several places needs to be combined, and
4. the number of elements in an N-tuple not necessarily equals 5 (e.g., for two-dimensional tables).

All this was addressed previously in the harvesters via (reduplicated) program code. Now, as the scraper delivers N-tuples, we can use *HFC* (see Section 3.) to mediate between the different representations. This is achieved by writing declarative mapping rules in *HFC* which take N-tuples as input and yield quintuples for the target ontologies. Here is an example dealing with *adidas*' foundation year. The scraper correctly delivers

```
scrap:adidas_AG_1382726391024
scrap:HEAD_Corporate_Information
scrap:Corporate_Information_1382726391026
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .

scrap:Corporate_Information_1382726391026
scrap:KEY_Established
"1949"
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .
```

The simple migration rule then reduces to

```
?comp scrap:HEAD_Corporate_Information ?ci ?s ?e
?ci scrap:KEY_Established ?year ?s ?e
->
?comp dax:foundedIn ?gyear ?s ?e
@action
?gyear = MakeGYear ?year
```

where the RHS action generates an XSD Gregorian year bound to `?gyear`, given the XSD string stored in `?year`, thus delivering the schema-compliant quintuple (sub-ontology: *DAX*)

```
scrap:adidas_AG_1382726391024
dax:foundedIn
"1949"^^xsd:gYear
"2013-10-25T20:39:51"^^xsd:dateTime
"2013-10-25T20:39:51"^^xsd:dateTime .
```

## 5. Ontology Alignment

As mentioned in Section 3.1., we are currently investigating methods to automatically relate the multilingual labels and definitions from *NACE*, *ICB*, *DAX*, and *GICS* with one another, but also with the free-text information from company instances (specified on their website) in order to establish a unified and better sector classification.

For instance, from the English info text found for *adidas*

> *The adidas Group is one of the global leaders within the sporting goods industry ...*

it should be feasible to find the class nace:nace_47.64 whose English label is

> *Retail sale of sporting equipment in specialised stores.*

For the example from Section 3.1. (*financial institutions*), the mapping is quite easy (at least for *ICB*, *DAX*, and *GICS*) as the English and German labels for the three ontology coincide (*banks* and *Banken*, resp.). Concerning the *NACE* concept nace_64.1, the mapping between its

German description *Zentralbanken und Kreditinstitute* and *Banken* is probably easier than it is for the English case (*NACE*: *monetary intermediation*). Thus it is very useful to have more than one language pair, and if in doubt, checking other parallel language labels, if this is possible.

Using pairs of natural language texts (in our case: labels and definitions for the concepts from the four ontologies) to decide entailments $T \models H$ between a *text* $T$ and a *hypothesis* $H$, methods from the *(recognizing) textual entailment* (RTE) research field are of great importance here; see (Chierchia and McConnel-Ginet, 2000; Dagan et al., 2006).

Some of the RTE approaches, e.g., (Wang, 2011), even consider a non-Boolean classification scheme, by assuming four textual semantic relations $e$, $p$, $c$, and $c$:

1. $T\,e\,H$ — $T$ *entails* $H$, $H$ is entailed by $T$
2. $T\,p\,H$ — $H/T$ is a *paraphrase* of $T/H$
3. $T\,c\,H$ — $T$ and $H$ are *contradictory*
4. $T\,u\,H$ — the relation between $T$ and $H$ is *unknown*

This classification scheme fits nicely with the open-world assumption in OWL. Assuming that $T$ and $H$ come from the info text (label, definition) for the industry sector classes $C$ and $D$, we are allowed to generate the following description logic axioms for the above four semantic relations:

1. $C \sqsubseteq D$
2. $C \equiv D$
3. $\bot \equiv C \sqcap D$
4. $\bot \sqsubseteq C \sqcap D$

Of course, the last axiom does not provide any more information in an open world as the relation between $T$ and $H$ is unknown at the time when the axiom was generated.

Computing the entailment relations between several language pairs for the same two concepts will result in description logic axioms with a higher degree of being correct. For instance, when assuming English and German labels $(T_E, T_G, H_E, H_G)$ for the same two concepts $C$ and $D$, from

$$T_E\,p\,H_E \;\&\; T_G\,c\,H_G$$

it is wise *not* to generate any description logic axiom involving $C$ and $D$; however,

$$T_E\,p\,H_E \;\&\; T_G\,e\,H_G$$

either leads to a *credulous* axiom

$$C \equiv D$$

or to a more *skeptical* one

$$C \sqsubseteq D$$

whereas the latter is always correct w.r.t. both textual entailments.

# 6. Further Issues

We utilize this final section to present some of our findings of what we believe are shortcomings of ontologies which adhere to mere binary ABox relation instances.

## 6.1. Tuple- Vs. Triple-Based Representations

In TRENDMINER, the integrated ontology as well as the complete ABox data is uploaded to the OWLIM semantic repository (Kiryakov et al., 2005), hosted by Ontotext, one of our partners in the project. Attentive readers of this paper, however, will ask themselves how this goes together with the representation of company snapshot data, as explained in Section 4. We demonstrated that snapshot data is encoded via quintuples, whereas ordinary semantic repositories (such as OWLIM or Virtuoso) always assume a triple-based representation.

In order to make the snapshots accessible in OWLIM, we perform a semantic-preserving quintuple-to-triple conversion which is compatible with W3C's *N-ary Relations* Best Practice proposal (Hayes and Welty, 2006). A description of further possible representation schemes can be found, e.g., in (Krieger et al., 2008). The idea behind the reduction is quite simple: all arguments lying in the range of a relation instance are hidden in a "container" object. The hidden arguments, in our case the actual value of the atemporal binary fact, the starting and the ending time can be obtained through pre-defined properties. Thus a quintuple

```
subj pred obj start end .
```

might equivalently be represented through 5 triples:

```
subj pred cont .
cont rdf:type nary:RangePlusTimeContainer .
cont nary:value obj .
cont nary:starts start .
cont nary:ends end .
```

Note that `cont`, the container object, is a brand-new individual, usually a RDF blank node, that needs to be introduced for each quintuple.

Given such a representation, we can now query useful information, say, the evolution of the *total capital stock* for *adidas*, the company on which we focussed in Section 4.:

```
SELECT ?v ?s
WHERE {
  ?c dax:isin ?i .
  ?i nary:value "DE000A1EWWW0" .
  ?c dax:totalCapitalStock ?t
  ?t nary:value ?v .
  ?t nary:starts ?s .
}
```

Compare this relatively easy question with a query that would directly operate on quintuples:

```
SELECT ?v ?s
WHERE {
  ?c dax:isin "DE000A1EWWW0" ?s ?e .
  ?c dax:totalCapitalStock ?v ?s ?e .
}
```

Not only are less clauses involved (5 vs. 2), but both the repository representation as well as the query involves more individuals, viz., the container objects bound to the variables `?i` and `?t`. The different representations not only result in different space requirements (only a constant factor of 2.2 during practical measurements), but can have a *massive* influence on the runtime performance and termination of the materialization process in a semantic repository: depending on the size and quality of data, querying & reasoning operate on a scale between *doable* and *intractable*; see (Krieger, 2012).

### 6.2. Going Beyond Binary Relations

The use of quintuples in *HFC* seems to indicate that the additional arguments are only needed to add a temporal extent to an atemporal binary relation instance. This is not true. Many seemingly binary relations (for the moment, we forget about the two temporal arguments) come with hidden arguments that we have not addressed in the specification of the relation, due to the fact that OWL in particular and description logics in general adhere to at most binary relations.

For instance, the binary relation obtains in the *BIO* ontology between Person and Degree is missing at least one further argument, viz., the educational organization (EduOrg) from which the degree was obtained. We can circumvent the problem by introducing a further binary property obtainedFrom defined on Degree, mapping to EduOrg. This modeling, however, loses the original connection to Person here, thus such an ontology is hard to read for a human. Clearly the relation composition

$$\mathsf{educatedAt} \equiv \mathsf{obtainedFrom} \circ \mathsf{obtains}$$

would help to obtain the educational institution, but such a definition is outside the expressive means of OWL. However, the following *HFC* rule is a direct encoding of the above composite relation:

```
?p bio:obtains ?d
?d bio:obtainedFrom ?eo
->
?p bio:educatedAt ?eo
```

Such a kind of specification is only possible in semantic repositories which offer a rule-based language (e.g., OWLIM or *HFC*) that is open to the user and do not come with (hidden) built-in rules, or even no rules at all. In order to address such critical relations, ontology/schema modeling has several choices:

1. use several relations and use rules, if possible (see above); example: educatedAt, obtainedFrom, obtains.

2. turn the relation into a class, expressing an event; example: Obtaining event, together with new properties, such as who, what, where, when.

3. be open to a direct encoding of arbitrary, possibly underspecified N-ary temporal relations; example in *HFC*: quinternary relation (sextuple) obtains $\subseteq$ $\mathsf{Person} \times \mathsf{Degree} \times \mathsf{EduOrg} \times \mathsf{dateTime}^2$.

We clearly opt for solution (3.) as it is natural, compact, intuitive, and less error prone.

## 7. Acknowledgements

The research described in this paper has been financed by the European Project TRENDMINER under contract number FP7 ICT 287863. The authors would like to thank our three reviewers for their encouraging and detailed comments. We would also like to thank our colleagues Ingrid Aichberger, Bernd Kiefer, Ashok Kumar, and Paul Ringler. Finally, we want to say a big *thank you* to the providers of the pivotal data on which our ontologies are based.

## 8. References

Chierchia, Gennaro and McConnel-Ginet, Sally. (2000). *Meaning and Grammar: Am Introduction to Semantics*. MIT Press, Cambridge, MA, 2nd edition.

Dagan, Ido, Glickman, Oren, and Magnini, Bernardo, (2006). *The PASCAL Recognising Textual Entailment Challenge*, pages 177–190. Lecture Notes in Computer Science. Springer.

Grant, Jan and Beckett, Dave. (2004). RDF test cases. Technical report, W3C, 10 February.

Hayes, Patrick and Welty, Chris. (2006). Defining N-ary relations on the semantic web. Technical report, W3C.

Hayes, Patrick. (2004). RDF semantics. Technical report, W3C.

Kiryakov, Atanas, Ognyanov, Damyan, and Manov, Dimitar. (2005). OWLIM – a pragmatic semantic repository for OWL. In *Proceedings of the International Workshop on Scalable Semantic Web Knowledge Base Systems*, pages 182–192.

Krieger, Hans-Ulrich, Kiefer, Bernd, and Declerck, Thierry. (2008). A framework for temporal representation and reasoning in business intelligence applications. In *AAAI 2008 Spring Symposium on* AI Meets Business Rules and Process Management, pages 59–70. AAAI.

Krieger, Hans-Ulrich. (2010). A general methodology for equipping ontologies with time. In *Proceedings LREC 2010*.

Krieger, Hans-Ulrich. (2012). A temporal extension of the Hayes/ter Horst entailment rules and an alternative to W3C's n-ary relations. In *Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS 2012)*, pages 323–336.

Krieger, Hans-Ulrich. (2013). An efficient implementation of equivalence relations in OWL via rule and query rewriting. In *Proceedings of the 7th IEEE International Conference on Semantic Computing (ICSC)*, pages 260–263.

Kumar Nedunchezhian, Ashok. (2013). Design and implementation of a model for syntactic scraping of structured data for creating RDF datasets. Master's thesis, Saarland University.

McGuinness, Deborah L. and van Harmelen, Frank. (2004). OWL Web Ontology Language Overview. Technical report, W3C, 10 February.

ter Horst, Herman J.˙ (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3:79–115.

Wang, Rui. (2011). *Intrinsic and Extrinsic Approaches to Recognizing Textual Entailment*. Ph.D. thesis, Saarland University.

Westerski, Adam, Iglesias, Carlos A. and Rico, Fernando Tapia. (2011). Linked opinions: Describing sentiments on the structured web of data. In *Proceedings of the 4th Workshop on Social Data on the Web*.